

bridging the gap

Daniel Jackson
MIT Lab for Computer Science
SDP · Vanderbilt · Dec 13, 2001

time to fulfill a promise

code alone is bad

- poor medium for exploring designs
- traps domain & design knowledge

design models alone are bad

- running system needs details
- if code diverges, design is useless

challenge

- bridge gap between design models & code

what kind of analysis?

at level of API & problem

- “no event listener cycles”, “no lost aircraft”

structure

- heap configurations & how they change

non-uniform

- critical properties, not critical programs

truly modular

- no whole-program assumption

unsound & incomplete

- inevitable cost of expressiveness

role of types

type systems gain soundness by

- uniformity: can't focus on crucial parts
- tractability: can't express complex invariants

two distinct benefits

- catching bugs (adding integer to string)
- decoupling code (rep independence)

so use types to

- modularize program for analysis
- type structure + property -> scope of analysis

why now?

languages are mature & simple

- programming (no address arithmetic anymore)
- modelling (no denotational semantics anymore)

algorithms for analysis are mature

- BDD's, bit-state hashing, SAT solvers

machines are fast

- 1980 : 2001 \approx 1 day : 1 second

API's are proliferating

- a Trojan horse, start with anomalies

relevant projects & position papers

Alur, Krishnamurthi, Smolka: model checking

Crary: more powerful types

Jackson: micromodels

Karsai: detailed design models

Larus: exploiting power on desktop

model checking of code

- Bandera, SLAM, Pathfinder, Fever

novel static analyses

- Parametric Shape Analysis, Role Analysis

code checkers

- Belief checking, ESC, JML, PREfix